# IML_project

Heikki Nenonen

2022-12-05

## Todo

- ~~dummy classifier~~
- ~~class4 -> event/nonevent, week1 exe?~~
- ~~drop partlybad, pelkkää FALSEa~~
- **varianssit mukana/ei mukana? ei one hot -> yksinkertaistaa liikaa ja tarkoitettu kategoriseen dataan**
- ~~date? paljon informaatiota, mutta halutaanko muuttujaksi <- opeta 2000-2008, testaa 2009-2011 / kysy slack test_hidden ei - - date, jätetäänkö pois? good riddance!~~
- ~~train, test, cv-10?~~
- ~~itse logisticregression, week2 exe1 <- lasso/ridge~~
- ~~accuracy, perplexity, week2 exe1~~
- accuracy of our accuracy? <– malli train+test, vähän parempi kuin pelkkä train?
- ~~class4 -> nonevent/1a/1b/II/ = 0,1,2,3~~
- googlaa mahdollisia malleja
- logreg/randomforest/qda with default parameters are best atm, maybe we can optimize
- plot

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model

#npf_test = pd.read_csv("initial_data/npf_test_hidden.csv")
npf_train = pd.read_csv("initial_data/npf_train.csv")
```

```
npf_train_test = npf_train.set_index("date")
npf_train_test = npf_train_test.drop(['id', 'partlybad'], axis=1)

class2 = np.array(["nonevent", "event"])
class2 = class2[(npf_train_test["class4"]!="nonevent").astype(int)]
#class2 = class2.apply(lambda x: 1 if "event" else 0)
npf_train_test.insert(loc=0, column="class2", value=class2)

npf_train_test["class2"].replace(["event", "nonevent"],[1,0], inplace=True)
npf_train_test["class4"].replace(["nonevent", "Ia", "Ib", "II"],[0, 1, 2, 3], inplace=True)
```

```
npf_train_test
```

```
##             class2  class4  CO2168.mean  ...  UV_B.std   CS.mean    CS.std
## date                                     ...
## 2000-01-17       1       2   368.771711  ...  0.018122  0.000243  0.000035
## 2000-02-28       0       0   378.197295  ...  0.003552  0.003658  0.000940
## 2000-03-24       1       2   373.043158  ...  0.272472  0.000591  0.000191
## 2000-03-30       1       3   375.643019  ...  0.451830  0.002493  0.000466
## 2000-04-04       0       0   377.661030  ...  0.291457  0.004715  0.000679
## ...            ...     ...          ...  ...       ...       ...       ...
## 2011-08-16       0       0   381.016623  ...  0.496816  0.002423  0.000425
## 2011-08-19       0       0   383.698146  ...  0.726461  0.002476  0.000902
## 2011-08-21       0       0   379.279128  ...  0.363890  0.003484  0.000457
## 2011-08-22       0       0   384.443758  ...  0.595032  0.004782  0.001082
## 2011-08-27       0       0   382.230839  ...  0.722553  0.006956  0.000605
##
## [464 rows x 102 columns]
```

```
#@ignore_warnings(category=ConvergenceWarning)
def loss(X_tr, y_tr, X_te, y_te, m):
    return mean_squared_error(y_te, m.fit(X_tr, y_tr).predict(X_te), squared=False)


def accuracy(X_tr, y_tr, X_te, y_te, m):
    return accuracy_score(y_te, m.fit(X_tr, y_tr).predict(X_te))

#def perplexity(p, y_test):
#    return np.exp(-np.mean(np.log(y_test*p + (1 - y_test) * (1 - p))))

#perplexity = lambda p: np.exp(-np.mean(np.log(y_test*p + (1 - y_test) * (1 - p))))
```

```
knitr::kable(py$results_class2, row.names = TRUE, digits = 2)
```

| | train_loss | cv_loss | test_loss | test_accuracy | test_perplexity1 |
|---|---|---|---|---|---|
| 1 | 0.71 | 0.71 | 0.71 | 0.49 | 2.00 |
| 2 | 0.34 | 0.35 | 0.33 | 0.89 | 1.38 |
| 3 | 0.38 | 0.39 | 0.36 | 0.87 | 1.44 |
| 4 | 0.38 | 0.39 | 0.36 | 0.87 | 1.44 |
| 5 | 0.34 | 0.35 | 0.33 | 0.89 | 1.38 |
| 6 | 0.38 | 0.39 | 0.36 | 0.87 | 1.44 |
| 7 | 0.43 | 0.43 | 0.41 | 0.83 | Inf |
| 8 | 0.00 | 0.40 | 0.36 | 0.87 | Inf |
| 9 | 0.00 | 0.33 | 0.29 | 0.91 | 1.29 |
| 10 | 0.36 | 0.44 | 0.39 | 0.85 | Inf |
| 11 | 0.26 | 0.35 | 0.33 | 0.89 | 0.00 |

```
knitr::kable(py$results_class4, row.names = TRUE, digits = 2)
```

| | train_loss | cv_loss | test_loss | test_accuracy | test_perplexity1 |
|---|---|---|---|---|---|
| 1 | 1.73 | 1.73 | 1.71 | 0.51 | 1.43 |
| 2 | 1.08 | 1.11 | 1.18 | 0.68 | 1.26 |

|    | train_loss | cv_loss | test_loss | test_accuracy | test_perplexity1 |
|---|---:|---:|---:|---:|---:|
| 3 | 1.13 | 1.14 | 1.11 | 0.70 | 1.26 |
| 4 | 1.14 | 1.14 | 1.22 | 0.68 | 1.26 |
| 5 | 1.08 | 1.11 | 1.14 | 0.70 | 1.26 |
| 6 | 1.14 | 1.14 | 1.22 | 0.68 | 1.26 |
| 7 | 1.38 | 1.38 | 1.30 | 0.64 | 26.42 |
| 8 | 0.48 | 1.73 | 1.71 | 0.51 | Inf |
| 9 | 0.00 | 1.03 | 0.85 | 0.72 | 1.22 |
| 10 | 1.01 | 1.23 | 1.24 | 0.53 | 13.88 |
| 11 | 0.92 | 1.08 | 1.09 | 0.72 | 0.00 |

```
#shows all columns
#required package tabulate
#print(res.to_markdown())
models = [
  RandomForestClassifier(criterion='gini'),
  RandomForestClassifier(criterion='log_loss'),
  RandomForestClassifier(criterion='entropy')]

results_class4 = magic(models, 'class4')
```

```
## /home/artkoski/.local/lib/python3.8/site-packages/pandas/core/arraylike.py:397: RuntimeWarning: inval
##   result = getattr(ufunc, method)(*inputs, **kwargs)
## /home/artkoski/.local/lib/python3.8/site-packages/pandas/core/arraylike.py:397: RuntimeWarning: inval
##   result = getattr(ufunc, method)(*inputs, **kwargs)
```

```
print(results_class4.to_markdown())
```

```
## |                                              |   train_loss |   cv_loss |   test_loss |   test_accu
## |:---------------------------------------------|-------------:|----------:|------------:|-----------
## | RandomForestClassifier()                     |            0 |   1.07903 |    0.743768 |        0.7(
## | RandomForestClassifier(criterion='log_loss') |            0 |   1.05909 |    1.01058  |        0.7:
## | RandomForestClassifier(criterion='entropy')  |            0 |   1.05896 |    0.887262 |        0.7(
```